Universidad Nacional de Colombia - Sede Bogotá
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Curso: Ingeniería de Software
Documentación builder

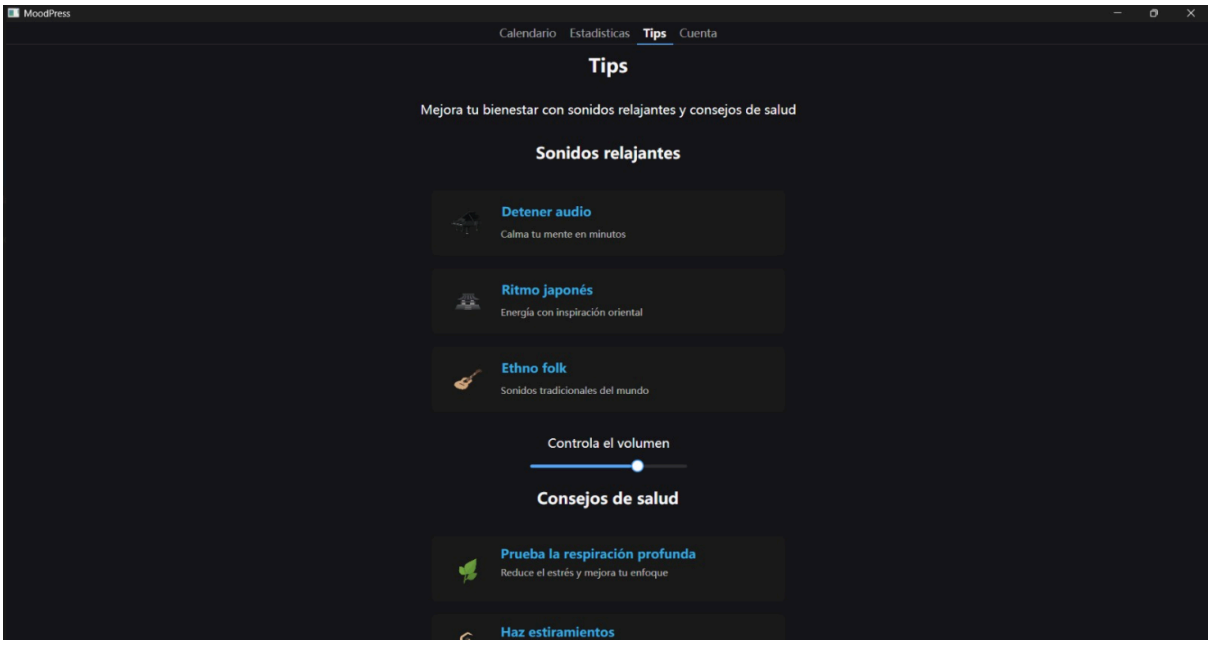
Grupo: Los Inadaptados

Estudiantes:

- Camilo Lagos Malaver
- John Freddy Morenazo Alejo
- Juan Felipe Marin Moreno
- Sergio Nicolas Escobar

Documentación builder

- En nuestro proyecto utilizamos el patrón de diseño **Builder** 🏗️, este es un **patrón creacional** 🔧 que nos permite crear diferentes representaciones de un objeto sin necesidad de usar constructores con demasiados parámetros 🧩, utilizando el mismo código de construcción 🛠️.
- En nuestro proyecto, builder fue implementado en la pantalla de tips



Pantalla de tips del proyecto

Como podemos observar, en la pantalla de **tips** 💡 hay diferentes **tarjetas** 🗂️, correspondientes a las tarjetas para **reproducir sonidos relajantes** 🎵🧘, y a las tarjetas de **tips de salud** ❤️‍🩹🍎.

- Implementamos **Builder** 🏗️ al ver que era redundante crear diferentes clases o llamar tantos parámetros en los constructores 🧩 (por ejemplo, la diferencia entre ambas tarjetas es que en una el título es un botón 🔘 y en la otra solo un texto 📝), y nos resultó mucho más sencillo y modular 🧩 (como cuando le quisimos agregar la posibilidad de que cada tarjeta tuviera una imagen 🖼️) usar este patrón de diseño.
- Podemos ver los detalles de la implementación de builder dirigiéndonos al commit respectivo

Project/src/application/components/CardBuilder.cs

```
@@ -0,0 +1,54 @@
+ using System;
+
+ namespace Project.presentation.components
+ {
+     public class CardBuilder
+     {
+         private readonly Card _card = new Card();
+
+         public CardBuilder WithAudioFile(string fileName)
+         {
+             _card.AudioFileName = fileName;
+             return this;
+         }
+
+         public CardBuilder WithTitle(string title)
+         {
+             _card.TitleText = title;
+             return this;
+         }
```

CardBuilder es el constructor, implementa la construcción paso a paso de cada card

Project/src/application/components/audioPlayer.axaml.cs

```
@@ -0,0 +1,53 @@
+ using Avalonia.Controls;
+ using Avalonia.Markup.Xaml;
+ using Project.application.components;
+ using System;
+
+ namespace Project.presentation.components
+ {
+     public partial class AudioPlayer : UserControl
+     {
+         private StackPanel _cardsContainer;
+
+         public AudioPlayer()
+         {
+             InitializeComponent();
+             CreateCards();
+         }
+
+         private void InitializeComponent()
+         {
+             AvaloniaXamlLoader.Load(this);
+             _cardsContainer = this.FindControl<StackPanel>("CardsContainer");
+         }
+
+         private void CreateCards()
+         {
+             // Create all cards with interactive (button) titles
+             var pianoCard = CardBuilder.CreateInteractive()  // Changed from CreateStandard to CreateInteractive
+                 .WithAudioFile("relaxingPiano.mp3")
+                 .WithTitle("Piano relajante")
+                 .WithDescription("Calma tu mente en minutos")
+                 .Build();
+
```

audioPlayer es una clase directora, orquesta y utiliza la construcción de cards

Project/src/application/components/card.axaml.cs

```
@@ -0,0 +1,215 @@
+ using Avalonia;
+ using Avalonia.Controls;
+ using Avalonia.Markup.Xaml;
+ using System;
+ using Project.presentation.components;
+
+ namespace Project.presentation.components
+ {
+     public partial class Card : UserControl
+     {
+         private TextBlock _titleTextBlock;
+         private Button _titleButton;
+         private ContentControl _titleContent;
+         private TextBlock _descriptionText;
+         private Audio _audioPlayer;
+         private bool _isPlaying = false; // Add state tracking
+
+         public static readonly StyledProperty<string> AudioFileNameProperty =
+             AvaloniaProperty.Register<Card, string>("AudioFileName");
+
+         public static readonly StyledProperty<string> TitleTextProperty =
+             AvaloniaProperty.Register<Card, string>("TitleText");
```

Card es el producto, el objeto final que se construye (la tarjeta)

También podemos revisar el commit de Health tips added para revisar el uso de builder en la creación de tarjetas relativas a tips de salud

Project/src/application/components/healthTips.axaml.cs

```
@@ -0,0 +1,82 @@
+ using Avalonia;
+ using Avalonia.Controls;
+ using Avalonia.Markup.Xaml;
+ using Avalonia.Media;
+
+ namespace Project.application.components
+ {
+     public partial class HealthTips : UserControl
+     {
+         private StackPanel _tipsContainer;
+
+         public HealthTips()
+         {
+             InitializeComponent();
+             CreateHealthTips();
+         }
+
+         private void InitializeComponent()
+         {
+             AvaloniaXamlLoader.Load(this);
+             _tipsContainer = this.FindControl<StackPanel>("TipsContainer");
+         }
```

healthTips, al igual que audioPlayer, es una clase directora, solo que ya no controla la creación de tarjetas de audio sino simplemente tarjetas con un título y descripción

Por último para revisar la modularidad de builder podemos dirigirnos al commit de icons added in tips, en el que se le añade la posibilidad de que cada tarjeta tenga imagenes

Project/src/application/components/CardBuilder.cs

```
@@ -1,5 +1,3 @@
- using System;
-
 namespace Project.presentation.components
 {
     public class CardBuilder
@@ -36,6 +34,13 @@ public CardBuilder WithTextTitle()
             return this;
         }

+        // Nuevo método para añadir imagen desde recursos
+        public CardBuilder WithImage(string iconResourcePath)
+        {
+            _card.ImageResource = iconResourcePath;
+            return this;
+        }
+
         public static CardBuilder CreateStandard()
         {
             return new CardBuilder().WithTextTitle();
```

En card builder basta con añadir un nuevo método para agregar la imagen

Project/src/application/components/audioPlayer.axaml.cs

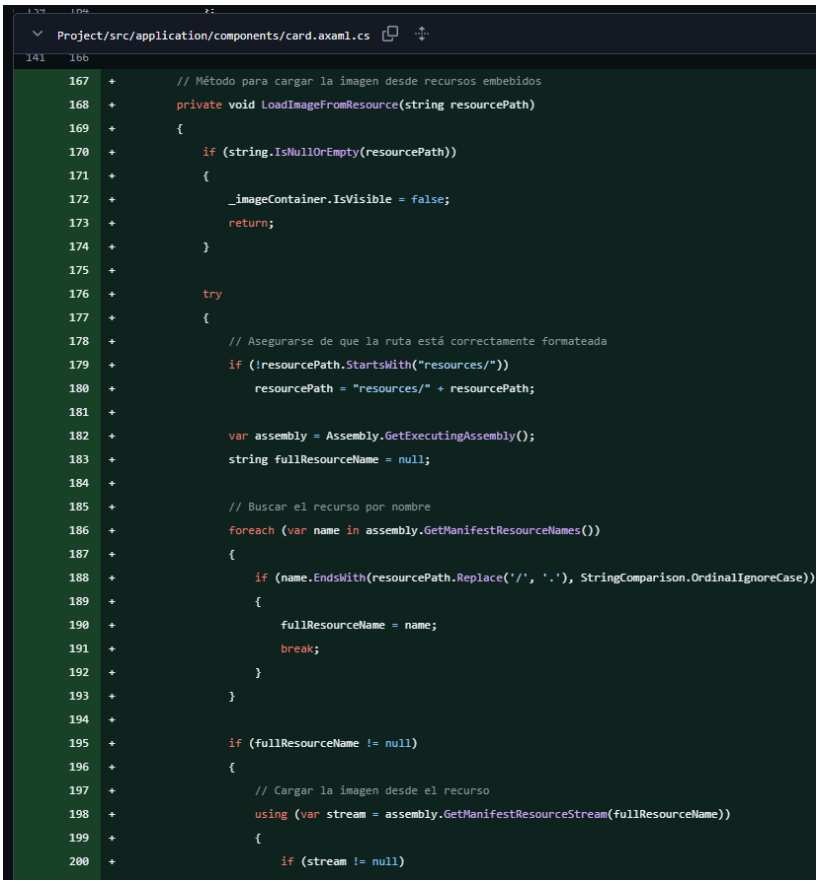
```
@@ -28,20 +28,23 @@ private void CreateCards()
                     .WithAudioFile("relaxingPiano.mp3")
                     .WithTitle("Piano relajante")
                     .WithDescription("Calma tu mente en minutos")
+                    .WithImage("icons/piano.png")
                     .Build();

                 // This one was already interactive
                 var japanCard = CardBuilder.CreateInteractive()
                     .WithAudioFile("japanBeat.wav")
                     .WithTitle("Ritmo japonés")
                     .WithDescription("Energía con inspiración oriental")
+                    .WithImage("icons/japan.png")
                     .Build();

                 // Changed this one from standard to interactive
                 var folkCard = CardBuilder.CreateInteractive()  // Chang
                     .WithAudioFile("ethnoFolk.wav")
                     .WithTitle("Ethno folk")
                     .WithDescription("Sonidos tradicionales del mundo")
+                    .WithImage("icons/guitar.png")
                     .Build();

                 // Add all cards to the container
```

En audioPlayer es suficiente especificar que se quiere que las tarjetas tengan imágenes, análogamente en healthTips



Project/src/application/components/card.axaml.cs

```
141  166
     167 +        // Método para cargar la imagen desde recursos embebidos
     168 +        private void LoadImageFromResource(string resourcePath)
     169 +        {
     170 +            if (string.IsNullOrEmpty(resourcePath))
     171 +            {
     172 +                _imageContainer.IsVisible = false;
     173 +                return;
     174 +            }
     175 +
     176 +            try
     177 +            {
     178 +                // Asegurarse de que la ruta está correctamente formateada
     179 +                if (!resourcePath.StartsWith("resources/"))
     180 +                    resourcePath = "resources/" + resourcePath;
     181 +
     182 +                var assembly = Assembly.GetExecutingAssembly();
     183 +                string fullResourceName = null;
     184 +
     185 +                // Buscar el recurso por nombre
     186 +                foreach (var name in assembly.GetManifestResourceNames())
     187 +                {
     188 +                    if (name.EndsWith(resourcePath.Replace('/', '.'), StringComparison.OrdinalIgnoreCase))
     189 +                    {
     190 +                        fullResourceName = name;
     191 +                        break;
     192 +                    }
     193 +                }
     194 +
     195 +                if (fullResourceName != null)
     196 +                {
     197 +                    // Cargar la imagen desde el recurso
     198 +                    using (var stream = assembly.GetManifestResourceStream(fullResourceName))
     199 +                    {
     200 +                        if (stream != null)
```

En card se realizan la mayoría de los cambios, añadiendo la lógica para cargar la imagen desde los recursos embebidos y mostrarla